Face Detection & Recognition

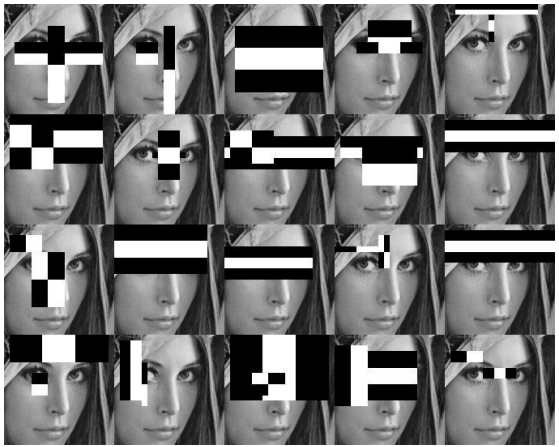
CS510 - Winter 2017
Final Project Presentation
Mike Lane

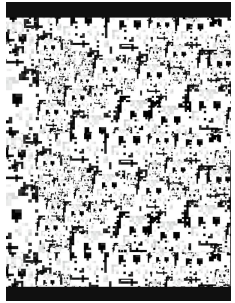
Face Detection & Recognition

- Classic problem in computer vision
- Many different applications
  - Security & Authentication
  - Law Enforcement
  - Identifying missing children
  - Organizing personal photos
  - Apps like Snapchat
  - Helping the blind
  - Etc.
- Easy and fast for humans; therefore, important for AI

Face Detection Using Haar Cascades

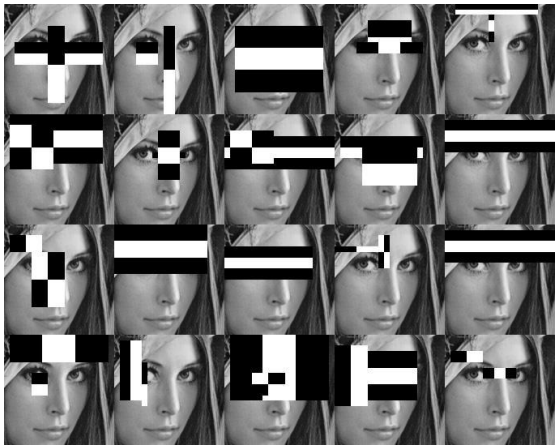
- Pros:
  - Built into OpenCV
  - Easy to implement
- Cons:
  - Very sensitive to hyperparameters
  - Quite easy to fool

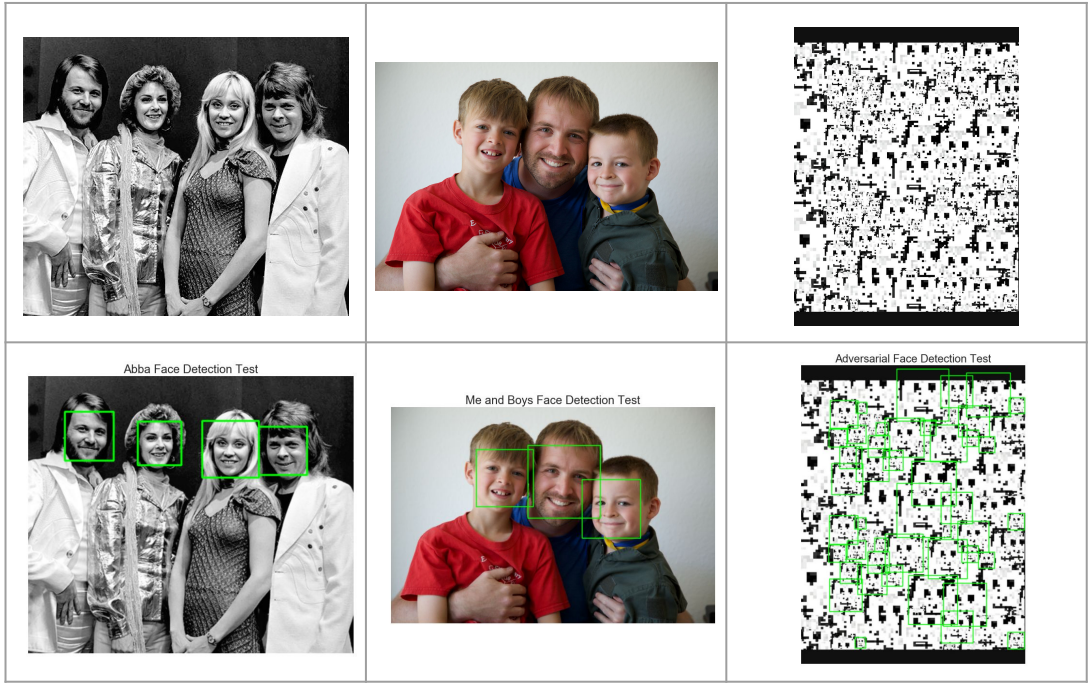


Face Detection Using Haar Cascades

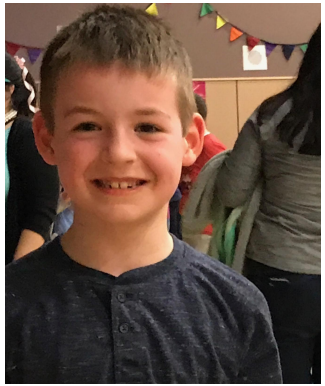
- Pros:
  - Built into OpenCV
  - Easy to implement
- Cons:
  - Very sensitive to hyperparameters
  - Quite easy to fool





Face Recognition: OpenFace

- Convolutional Neural Networks are the current state of the art
  - High accuracy
  - Long training times
  - Lots of data required to train
- Network can be trained and used for classification later
- OpenFace is a pre-trained CNN
- Super easy to implement as a Docker container



example training images

example testing images

Face Recognition: OpenFace

Face Recognition: OpenFace – Results

```
=== testing-images/Joshua-Lane-2.jpg ===
Predict joshua-lane with 0.65 confidence.

=== testing-images/Joshua-Lane-3.jpg ===
Predict joshua-lane with 0.80 confidence.

=== testing-images/Michael-Lane-1.jpg ===
Predict michael-lane with 0.67 confidence.

=== testing-images/Michael-Lane-2.jpg ===
Predict michael-lane with 0.73 confidence.

=== testing-images/Rebecca-Brown-1.jpg ===
Predict rebecca-brown with 0.85 confidence.

=== testing-images/Rebecca-Brown-2.jpg ===
Predict rebecca-brown with 0.86 confidence.

=== testing-images/William-Lane-4.jpg ===

=== testing-images/William-Lane-5.jpg ===
Predict william-lane with 0.84 confidence.
```

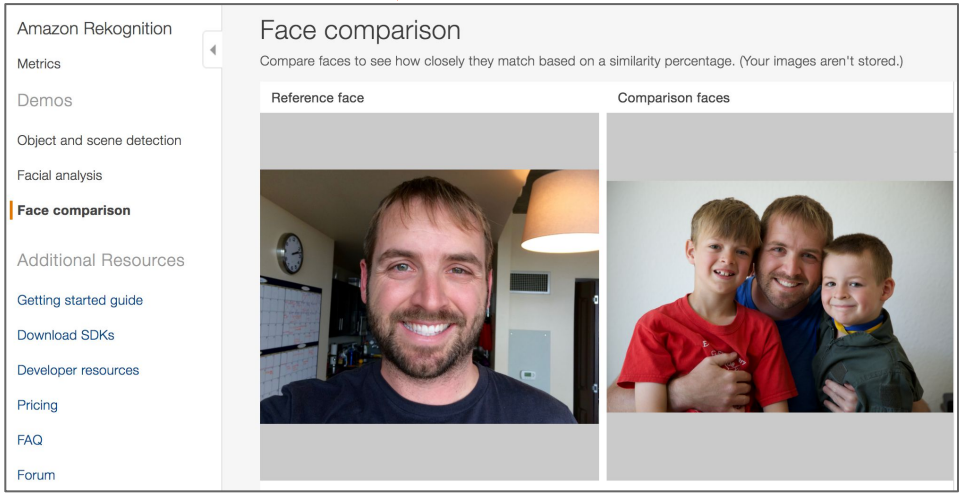
Face Recognition: OpenFace – Results

```
=== testing-images/Joshua-Lane-2.jpg ===
Predict joshua-lane with 0.65 confidence.

=== testing-images/Joshua-Lane-3.jpg ===
Predict joshua-lane with 0.80 confidence.

=== testing-images/Michael-Lane-1.jpg ===
Predict michael-lane with 0.67 confidence.

=== testing-images/Michael-Lane-2.jpg ===
Predict michael-lane with 0.73 confidence.

=== testing-images/Rebecca-Brown-1.jpg ===
Predict rebecca-brown with 0.85 confidence.

=== testing-images/Rebecca-Brown-2.jpg ===
Predict rebecca-brown with 0.86 confidence.

=== testing-images/William-Lane-4.jpg ===
???

=== testing-images/William-Lane-5.jpg ===
Predict william-lane with 0.84 confidence.
```

Face Recognition: AWS Rekognition

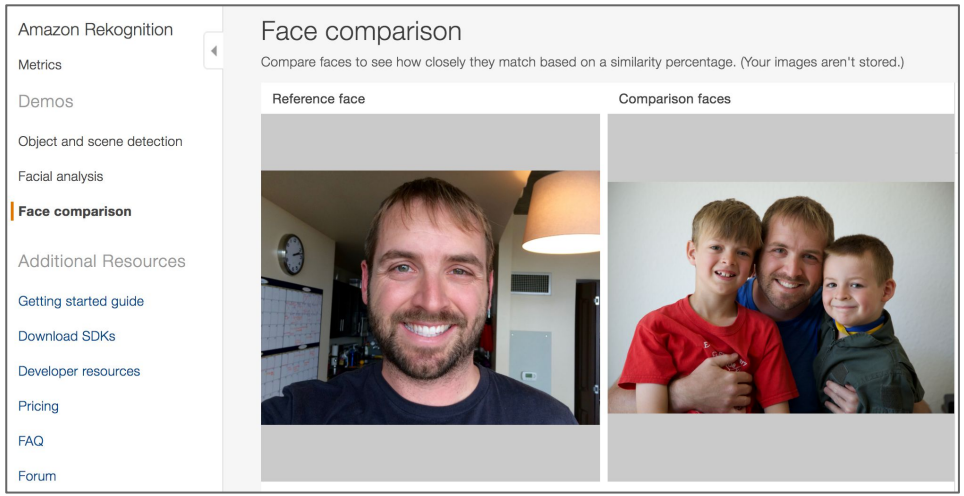
- Amazon is smart to get into the game
- Rekognition is one of the newest services offered as a part of AWS
- Make it easy for people and businesses to add face detection for things like security
- Any of us could implement a robust face recognition home security system!
- Backend is an opaque deep neural network, very little details available
- Also handles object detection and face detection

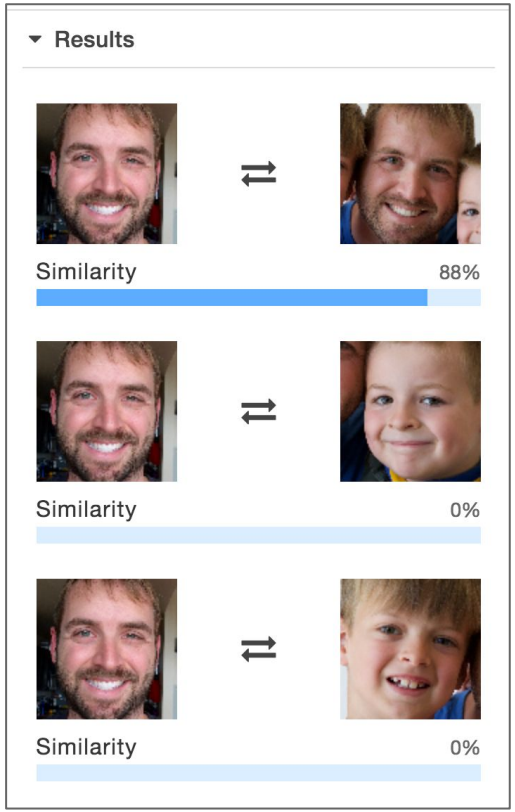
web-based demo interface



Face Recognition: AWS Rekognition

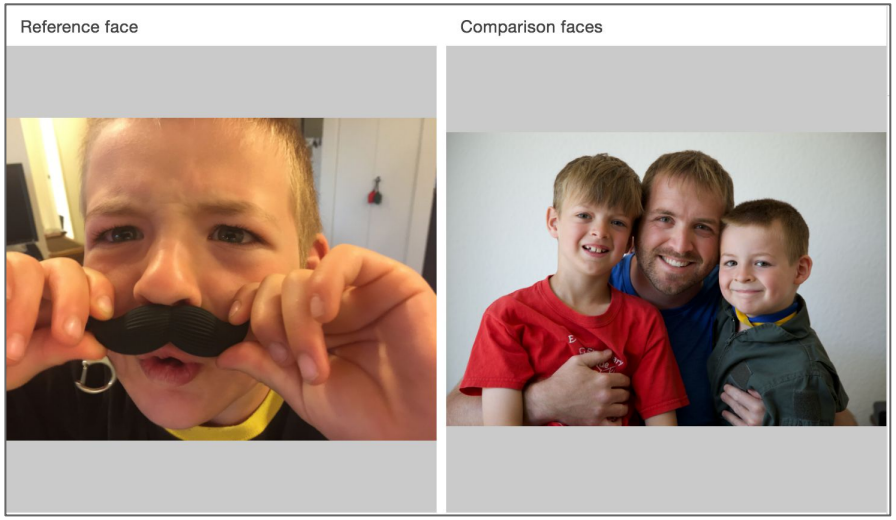
the results are often pretty good ...

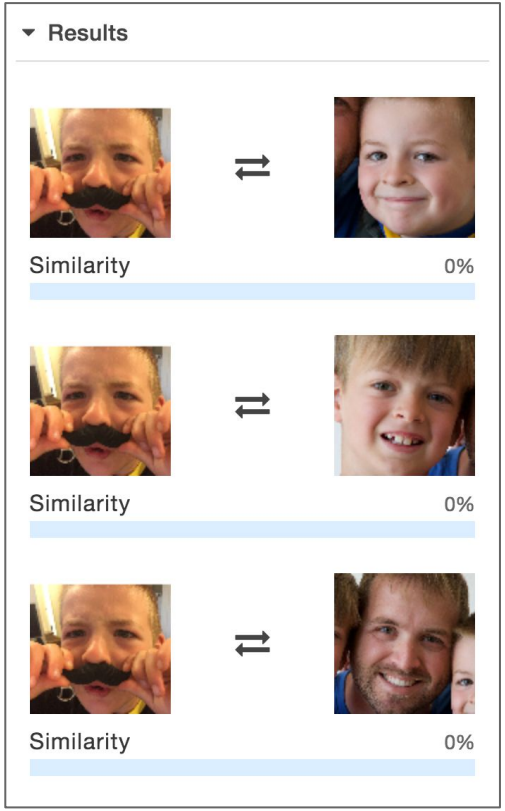




Face Recognition: AWS Rekognition

... but not always





Face Recognition: Scikit-Learn

- Scikit-Learn is a machine learning library for Python
- It has dozens of classifiers and regressors, supervised & unsupervised learning algorithms, and classifier evaluation tools
- There is even dataset download methods that let you download many well known datasets
- I modified a face recognition example in the documentation and I was able to have 7 face recognition algorithms implemented in no time

some examples from the labeled faces in the wild dataset

Face Recognition: Sklearn – Overview

1. Use `sklearn.datasets.fetch_lfw_people` to get the data
2. Use `sklearn.model_selection.train_test_split` to split the data into training and testing sets.
3. Use `sklearn.decomposition.PCA` for dimensionality reduction ala eigenfaces
4. Train ***one of the many classifiers*** that sklearn offers on the training data set
5. Use `sklearn.metrics.classification_report` and `sklearn.metrics.confusion_matrix` to evaluate the classifier

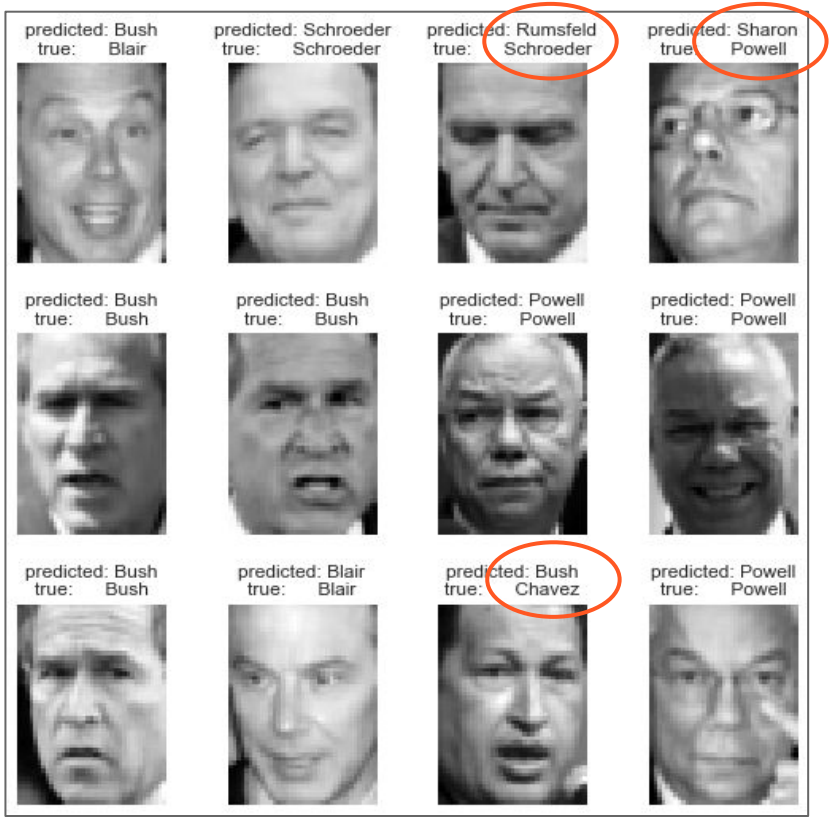
this example uses a support vector machine classifier

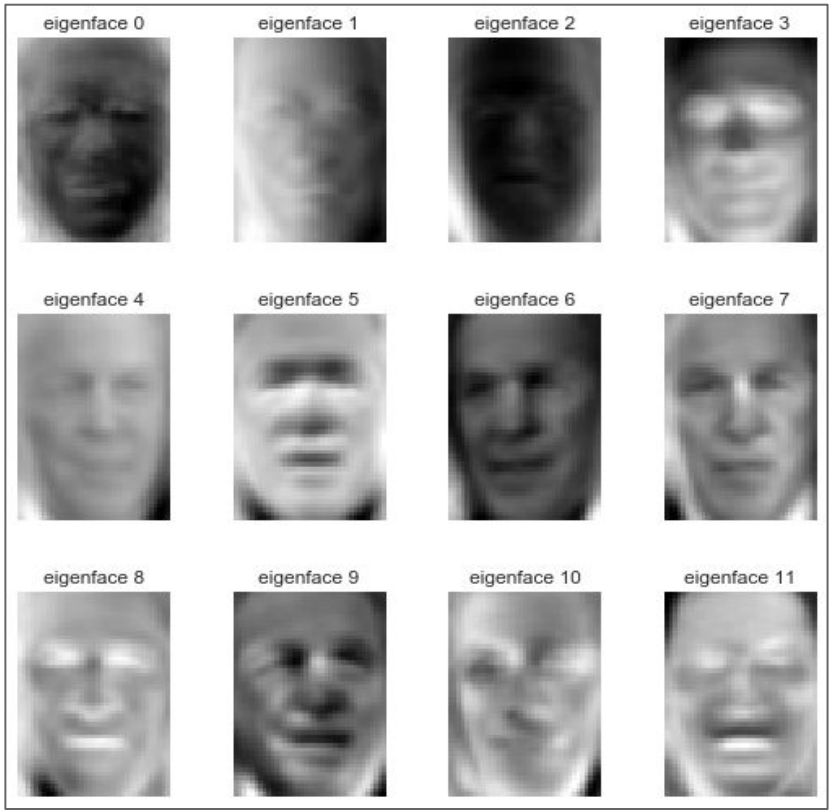
```
lfw_people = fetch_lfw_people(
      min_faces_per_person=70,
      resize=0.4)
### snip ###
X_train, X_test, y_train, y_test =train_test_split(
      X, y, test_size=0.25,
      random_state=np.random.RandomState())
### snip ###
pca = PCA(n_components=n_components,
          svd_solver='randomized',
          whiten=True).fit(X_train)
### snip ###
eigenfaces = pca.components_.reshape((n_components, h, w))
### snip ###
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
### snip ###
clf = GridSearchCV(
SVC(kernel='rbf', class_weight='balanced'), param_grid)
clf = clf.fit(X_train_pca, y_train)
y_pred = clf.predict(X_test_pca)
### snip ###
print(classification_report(y_test, y_pred,
target_names=target_names))
print(confusion_matrix(y_test, y_pred, labels=range(n_classes)))
```

Face Recognition: Scikit-Learn – Results

Classifier	Precision	Recall	F1
Support Vector Machine	0.85	0.85	0.85
Voting Classifier	0.85	0.84	0.84
Multilevel Perceptron	0.84	0.84	0.84
Logistic Regression	0.84	0.83	0.83
Stochastic Grad Desc	0.79	0.78	0.78
Gradient Boosting	0.63	0.63	0.60
Adaboost	0.51	0.52	0.49

Example Classifier Output Gallery





Questions?